Using model checking in the design of a sensor network protocol

Ivaylo Valkov Alice Miller

University of Glasgow

Abstract: We investigate how the PRISM and Alloy model checkers can be used in the design of a sensor network communication protocol. We introduce the two model checkers and illustrate how Alloy can be used to specify and analyse an existing communication protocol WirelessHART. We then propose how PRISM and Alloy will be used in the *design* of a new protocol, Ctrl-MAC, which is currently being developed as part of an EPSRC funded program grant, S4: Science of Sensor System Software. The aim is to exploit the strengths of each approach to allow us to select parameters and configurations to optimise the protocol.

1 Introduction

Testing is the most commonly used method for validation of software systems. However testing alone can not provide guarantees of complex system behaviour. Concurrent systems, such as communication protocols, are particularly hard to verify using testing. In such systems we want to prove temporal properties such as: "when a message is sent it will eventually arrive at its destination", or "if a message is sent from a component then it will receive an acknowledgement before the timeout period has elapsed".

Formal methods are commonly used for the verification of software and hardware systems. They comprise a range of techniques based on mathematics and logical reasoning, and are important in the creation of more robust and reliable systems. One such technique is model checking. We propose using model checking in the *design* of a new sensor network protocol. We can identify and prove properties of the protocol as it is developed - and adjust parameters accordingly. This differs from the common use in which properties of an existing protocol are verified, with no option to modify the protocol.

2 Model checking

Model checking is the process of creating a formal model of a software or hardware system and then using a software tool, called a model checker, to automate the search for proofs of or counterexamples to some properties of the system. The syntax and structure of the model that is being created depends on the choice of model checker, as each model checker relates a model to the underlying logical reasoning and logical concepts in a different way.

PRISM [5] is a probabilistic model checker that allows for the verification of a number of Markov chain variants, like Discrete Time Markov Chains (DTMCs) and Markov Decision Processes (MDPs). It has been used to formally verify quantitative properties of many network protocols including the device discovery phase of Bluetooth [2] and the CSMA/CA mechanism of the 802.15.4 based Zigbee standard [3].

The Alloy Analyzer (Alloy) is a model checker that uses a simple and powerful first-order logic language for specifying models which are then analysed with off-the-shelf SAT solvers. This allows models to be created in an iterative and incremental manner: they can be verified, inspected, evaluated and modified during multiple iterations. Furthermore, Alloy allows the configuration of the setting on which properties are being verified to be easily changed. For example, in the context of protocol analysis, models are often confined to a small fixed number of devices, which are placed in a particular configuration which should best exhibit the property under verification. Using Alloy a family of configurations can be analysed simultaneously.

Alloy has been used in the past to provide formal proofs for a variety of network protocols and to find security flaws in others. For example it has been used to formally verify five web security mechanisms that relate to user-supplied information [1]. In [4] Alloy is applied directly to model web protocols in a novel security analysis technique. In Section 3 we illustrate the use of Alloy for modelling an existing wireless protocol and in Section 4 we propose its use, along with PRISM, in the design of a new protocol.

3 An example: WirelessHART

We have investigated the use of Alloy for protocol analysis within the context of an existing protocol, namely the WirelessHART protocol, based on the IEEE 802.15.4 protocol standard.

WirelessHART is a short-range network protocol whose main goal is to perform low-cost communications over a network in such a way as to preserve battery life. It is a centralised protocol with one device acting as the personal area network (PAN) coordinator for the network. The protocol distinguishes between reduced function devices (RFDs), that are only able to gather and send data, and full function devices (FFDs), that are capable of transferring data from other nodes. All of the data is gathered at the PAN coordinator, which must be an FFD. Fig 1 shows an example of such a network.

As an illustration, we present below a small fragment of Alloy in which we declare the basic entities (atoms) that will be used in the model. These are referred to as signatures (*sig*).

```
// There isn't a device that is not
// a RFD or FFD
abstract sig Device { }
// Devices are either reduced function
// or full function
sig RFD, FFD extends Device { }
```

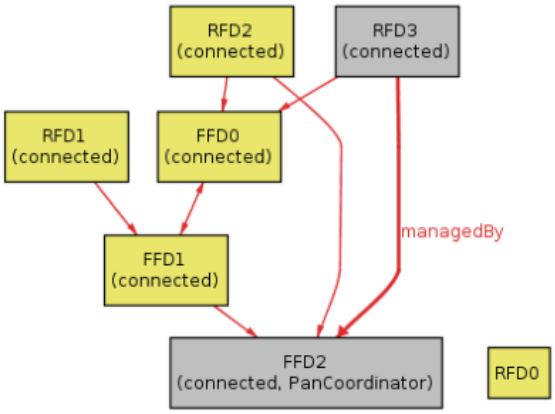

Figure 1: Network consisting of 3 FFDs and 4 RFDs. One device is not connected to the network.

After specifying the types of devices that exist we need to define how they relate in a network. To do so we create a Network signature which contains a number of nested relations. The *connected* relation is used to specify which devices belong to a given network. This is particularly useful when modelling scenarios where devices join or leave a network.

```
sig Network {
  connected: set Device,
  PanCoordinator: FFD & connected,
  managedBy: (connected - PanCoordinator)
                -> (FFD & connected),
  // RFDs cannot receive data
  receiveFrom: FFD -> Device
}{
  // connected devices are reachable
  // from the PanCoordinator
  connected in PanCoordinator.*receiveFrom
  // managedBy: inverse of
  // receiveFrom
  managedBy = ~receiveFrom
}
```

The *PanCoordinator* relation is used to specify a single device which acts as a central device for the network. The relations: *managedBy* and *receiveFrom* denote immediate connections between two devices. Finally, the *connected* relation defines that to be connected to a network a node means to be reachable from the central node.

In order to ensure that devices cannot send data to themselves, we add an additional constraint to the model:

```
fact {all nw: Network | all d: Device |
        d->d not in nw.receiveFrom }
```

We can now use Alloy to generate an instance of this model for a defined number and type of devices. Fig 1 is an example of such an instance. Manual inspection demonstrates that there are no self-related devices and that connected devices are appropriately marked.

4 Model checking for sensor network protocol design

PRISM is an obvious formalism for modelling communication protocolols and our wirelessHART example demonstrates the suitability of Alloy in this context. We propose to model and analyse a wireless communications protocol that is currently under development, in both PRISM and Alloy. Ctrl-MAC is a sensor network communication protocol that is being developed as part of the Science of Sensor Systems Software (S4) project. This is an EPSRC-funded project held by the University of Glasgow with the Universities of St Andrews and Liverpool and Imperial College. Ctrl-MAC is similar to WirelessHart in that they both use time division multiple access governed by a central gateway node. Its main goal is to provide reliable communication within a given time constraint for Cyber Physical Systems (CPS) such as water distribution systems and electric grids. By using model checking throughout the development of the protocol we will inform its design by choosing parameters and configurations to optimise performance.

References

[1] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song. Towards a formal foundation of web security. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 290–304, 2010.

[2] M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A formal analysis of Bluetooth device discovery. *Int. Journal on Software Tools for Technology Transfer*, 8(6):621–632, 2006.

[3] M. Fruth. *Formal Methods for the Analysis of Wireless Network Protocols*. PhD thesis, Oxford University, 2011.

[4] A Kumar. A lightweight formal approach for analyzing security of web protocols. In *Proc. RAID 2014*, pages 289–298.

[5] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV 2011*, pages 585–591.